

# KAOS

For People Who Have Got Smart

HARDWARE .. .. . DAVID ANEAR  
SOFTWARE .. .. . JEFF RAE  
FORTH .. .. . DAVID WILSON  
AMATEUR RADIO .. .. . ROD DRYSDALE VK3BYU  
EDUCATION .. .. . JEFF KERRY  
LIBRARY .. .. . RON KERRY  
TAPE LIBRARY .. .. . JOHN WHITEHEAD  
DISK LIBRARY .. .. . WARREN SCHAECH (B.H.)  
NEWSLETTER .. .. . IAN EYLES  
SYM .. .. . BRIAN CAMPBELL  
SECRETARY .. .. . ROSEMARY EYLES

OSI	SYM	KIM	AIM	UK101	RABBLE 65
-----	-----	-----	-----	-------	-----------

Registered by Australia Post  
Publication No. VBG4212

ADDRESS ALL CORRESPONDENCE TO 10 FORBES ST., ESSENDON, VIC. 3040

Vol.4 No.4

January 1984

We had a letter from Ed Richardson (SUPERBOARD) this week asking us to remind those members who complained last year that there was not a section in his competition for 32x64 format, that this year there is, and he is still waiting for their entries. On page 4, Ed says that he has 3 entries for the competition, but at the time of writing this has increased to 7 for the 24x24 and 1 for the 32x64 sections. Ed has extended the closing date, so get busy and get your entries in.

There will be a side-walk sale at the February meeting, so get rid of all that junk you bought last time - and buy some more. The sale will start at 1pm and the usual rules apply - you must make sure that anything you bring or buy is not left at the school.

In KAOS Vol.2 No.9 we asked for information about hardware and software that members had for sale so that we could print a list of what was available. We got no response at the time and now John Whitehead is going to try again, see page 15 and contact John if you can help.

Membership fees are now due. Only financial members will receive the February newsletter .

Meeting dates this year will be:

29th January	26th February	25th March	29th April	27th May	24th June
29th July	26th August	30th September	28th October	25th November	

The school children will be at the meetings in March, June and September.

The next meeting will be at 2pm on Sunday 29th January at the Essendon Primary School which is on the corner of Raleigh and Nicholson Streets, Essendon. The school will be open at 1pm.

The closing date for articles for the next newsletter is 10th February.

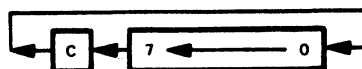
## INDEX

Competition .....	4	NOS Basicode .....	4
Civil War .....	5	OSI History pt 2 .....	13
For Sale .....	15	SB / Rabble Differences .....	3
For Sale ?? .....	15	RAM at \$C800 - Correction .....	5
Forth - Floating Point .....	11	Ratbas - Listing .....	6
I/O Bus - 16 Pin .....	8	Software Source .....	4
Machine Language pt 19 .....	2	SUPERBOARD .....	4

THE BEGINNING MACHINE LANGUAGE PROGRAMMER.....part 19  
*by David Dodds*

This month we will introduce 2 new instructions utilising accumulator addressing which was introduced last month. Both these instructions make use of the Carry flag in the status register as a ninth bit in the operation.

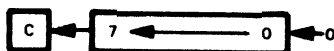
The first instruction is ROTATE LEFT (through Carry) or as a mnemonic ROL. A ROL is the 6502 version of musical chairs! It can use the accumulator as the operand but can also be applied to memory. During each ROL operation all bits in the data are moved one place to the left from the least significant towards the most significant value (see diagram below). The value in Bit 7 (MSB) moves to the Carry flag and the former value in Carry moves into Bit 0 (LSB).



Since the Carry takes part in the operation it must be affected. The negative (N) and zero (Z) flags are also updated during the operation. Since the ROL forms a closed loop system none of the bits are lost, only transplaced.

```
eg  A=1  00000001  C=1
after ROL A (Accumulator mode)
    A=3  00000011  C=0
after another ROL A
    A=6  00000110  C=0
```

The second instruction ASL for ARITHMETIC SHIFT LEFT is in some ways similar to the ROL. The ASL uses the same addressing modes as the ROL and data bits also move left one place. Unlike the ROL after each ASL the most significant bit (MSB) is lost. This happens because as the value in Bit 7 is moved to the Carry, a 0 is moved into the least significant bit (LSB) Bit 0.



The carry (C), N and Z flags are affected by an ASL.

```
eg  A= 10000101  ($85)  C=0
after ASL A
    A= 00001010  ($0A)  C=1
after another ASL A
    A= 00010100  ($14)  C=0
```

Note that an ASL is in effect a 2\* multiplication  
ASL and ROL are in fact commonly used together to form multiple precision multiplication routines.

A totally different application can be seen in the routine at \$FEDA in the machine language monitor of OSI machines.

The routine at this location is responsible for converting a Hexadecimal digit in ASCII format into four binary digits. This value is then rolled into the least significant four bits of the address field. On entering the routine the ASCII character is in the Accumulator and the X register contains the value two. A disassembly of the ROM reveals

```
FEDA LDY #4
FEDC ASL A      ;move bits 0 to 4 into bits 5 to 7
FEDD ASL A      ;to mask off ASCII value and prepare
FEDE ASL A      ;to roll into memory.
FEDF ASL A
FEE0 ROL A      ;Bit 7 to carry
FEE1 ROL $FC,X  ;Carry into Bit 0 of Address Low byte
FEE3 ROL $FD,X  ;Bit 7 from last ROL into Bit 0 Address High byte
FEE5 DEY
FEE6 BNE $FEE0  ;Do four times.FEE8 RTS
```

Next month we will look at two instructions which work exactly the opposite to ASL and ROL.

---

SOME RABBLE 65 AND SUPERBOARD DIFFERENCES  
*by John Whitehead*

From information supplied to Bill Roberts by Ray Gardiner.

	SUPERBOARD with DABUG III -----	RABBLE 65 -----
SCREEN CLEAR	PRINT CHR\$(127)	CLRS
	JSR \$FCD5	JSR \$FFD0
Scan keyboard and wait for a keypress	POKE11,0:POKE12,15*16+13 X=USR(X):K=PEEK(531) PRINT CHR\$(K)"="K	POKE11,0:POKE12,15*16+13 X=USR(X):K=PEEK(50228) PRINT CHR\$(K)"="K
(somewhere on the screen)	JSR \$FD00 STA \$D3CD	JSR \$FD00 STA \$D3CD
Scan keyboard without waiting (DABUG 3J only)	POKE11,2*16+1 POKE12,15*16+13 X=USR(X):K=PEEK(531) PRINT CHR\$(K)"="K  JSR \$FD21 STA \$D3CD	POKE11,13*16+12 POKE12,15*16+15 X=USR(X):K=PEEK(50228) PRINT CHR\$(K)"="K  JSR \$FFDC STA \$D3CD
Screen flip	24 x 24 POKE 251,0  48 x 12 POKE 251,1	64 x 32 POKE11,211:POKE12,255 X=USR(X)  80 x 25 POKE11,13*16+6:POKE12,255 X=USR(X)

# Superboard

January, 1984.

Newsletter of the Ohio Superboard User Group, 146 York Street, Nundah, 4012.

---

This month, I have set aside the continuation of the "Adventuring" series, in the hope that I will be able to list a small adventure, and then explain the operation of relevant routines. I am sure that you will find this approach to be more informative than just describing how to go about preparing the routines without any practical examples.

May I echo the call by David Dodds for more articles from the vast majority of our membership who never write for KAOS. Some of the most informative articles have resulted from a simple query, eg the Correction to Rom Basic in last months issue.

Although I would like to publish another twelve interesting SUPERBOARD editions this year, I know that I will run out of material by April or May. If anyone has any ideas for articles, I would be pleased to receive them.

There was no response to the Star Shoot problem-solving challenge in the October KAOS. Maybe nobody is interested in this use for a computer, or perhaps it was just too much trouble to type it in. For the time being, I will persevere and present some practical problems and applications in future issues of SUPERBOARD. I would have expected some interest from schools in what I regard as one of the few real uses for a home computer.

---

## JANUARY PROGRAMMING COMPETITION

It is early days yet, of course, but at the time of writing, I have just three entries in the competition. This will be the last one I run, as interest has steadily declined of late. Anyway, in the hope of not going out on a failure, I will extend the deadline for entries to January 31st, 1984. Chief advantages of entering are a crack at up to \$80 in prizemoney plus getting a lot of free software. This is your last chance to have a go!

---

## NOS BASICODE

In June, '82, I reported to members the existence of a proposed standard cassette format for all personal computers. Developed from the Hobbyscoop program of Radio Netherlands, the aim of NOS BASICODE is to provide a way for all computer owners to swap programs of a limited nature. Details are available of the protocol for the format, and also include conversion programs for various machines, including OSI, PET, APPLE, SORCERER, TRS-80, and Acorn Atom (forerunner of the BBC). I will be presenting the M/C listing of the OSI version in a future KAOS. I can also supply copies of the relevant part of the manual for any of the machines listed above, if you send 5 x 30c stamps to cover the cost of copying plus postage.

---

## UNEXPLORED SOFTWARE SOURCE

I recently received a letter asking for a full list of current software sources for the OSI. Unfortunately, I only know those which I have given in reviews. However, here is a source which is relatively unexplored:- Premier Publications, 208 Croydon Road, Anerley, London SE20 7YX has a couple of pages of listed software to suit both UK101 and C1P in various formats including 24 x 48. If you would like a copy, you can either write to them, or send 2 x 30c stamps to OSUG to cover photocopies and postage.

# — SUPERBOARD —

*Correction - RAM AT \$C800 ON RABBLE BOARD by Bernie Wills.*

I have found a problem with the installation of the 6116 as published in KAOS 3/8 page 7. The original installation had shown no signs of a fault during use over several months, and always passed a modified version of the OSUG M/C memory test. It failed as follows:-

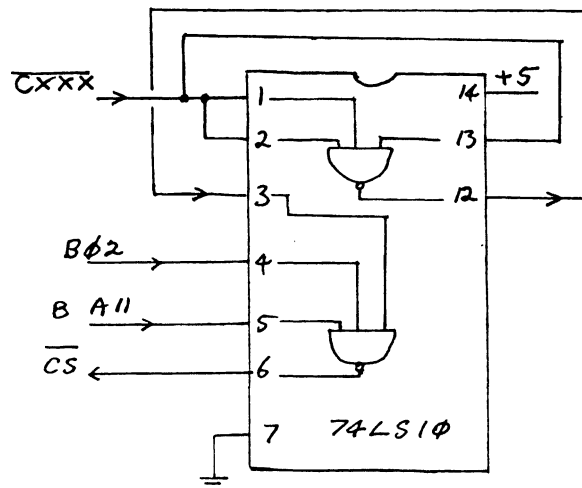
```
$C800 A9 00 LDA #$00
$C802 8D 20 C8 STA $C820
$C805 4C 00 FE JMP $FE00 ; to monitor.
```

A simple piece of code, but it repeatedly wrote a zero to \$C805, which was naturally interpreted as a Break instruction, usually causing a hangup.

Eventually, I realised that the decoding of the chip select for the 6116 did not use the 02 clock. There are good reasons for leaving 02 off the LS138 that gives the 4k decoding for \$8000+, as most chips are either eproms or I/O devices which have 02 connections. The 6116 needs it, however.

A simple solution is as follows:-

Replace the LS00 with a LS10 and rewire as the diagram below shows. Pins 1,2,5 and 6 are the same as in the Rabble manual, and the buffered 02 is taken from IC 6 pin 7, or from IC 10 pin 6.



## SOFTWARE REVIEW - Civil War

Civil War is a simulation game, the object being to win battles fought in the American Civil War. The player takes the part of the Confederacy. You are told the number of men you have and your budget, and also have these figures for the Yankees. You are told some very sketchy details of a conflict, and have first to allot your budget to food, salaries, and ammunition. You are then notified of the state of morale of your troops and whether you are attacking or defending a position. You have a choice of manoeuvres.

The program then processes this information somehow and comes up with the number of casualties and desertions, and if these figures are greater or less than the actual casualties in the real battle.

Depending on these factors, you win or lose the battle and move on to the next one.

First remove line number 491 else you only need to fight 4 battles to win 8.

I was rather disappointed with the game. It seemed childishly simple to win even for one who knew nothing of the Civil War. The game is a sort-of poor attempt to make the Civil War into a Hamurabi-type situation.

Perhaps someone with an interest in this kind of simulation would like to attempt a fix. The program is on the OSI label, and is in the Library.

by Michael Lemaire

[illegible]



## 16 PIN I/O BUS

*by David Tashner*

The 16 Pin I/O Bus concept, developed by OSI, is a terrific idea for interfacing a wide variety of peripheral devices. There is an absolute minimum requirement for the number of connecting wires. Fig 1 shows that there are only 16 lines needed, these can be connected via a 16 pin DIP header. The lines include an 8 Bit Data Bus, +5 and 0 volt connections. Read/Write (R/W) and phase 2 clock (O2) and 4 address lines, A0, A1 and LA2, LA3. On the Rabble 65 Computer and a number of plug in cards, the 16 pin socket is replaced with an 18 pin socket. This allows for two spare pins (9 and 10) for special functions.

Unlike some manufactures OSI do not go to much trouble to explain the whys and hows of their hardware. But to their credit the circuit diagrams for the complete range of Challenger computers are provided. On the OSI computers you will only find the 16 pin I/O bus on a computer with the Model 505 Rev B Floppy/processor board or those computers that have fitted the optional CA-20 8 port I/O Bus interface board. For those of you that have neither of these boards there are a number of 16 pin I/O adapters available.

This and subsequent articles will provide information on the inner workings of the 16 pin I/O bus and number of interfacing ideas that can be implemented on the bus. A complete circuit is given in Fig 2 to enable you to implement the Bus on your computer. You are not restricted to owning an OSI as the information supplied here can be used on just about any micro.

Fig 2 shows the hardware to implement the 16 pin Bus in its simplest form. This allows for a number of devices to be selected within the lowest 16 Bytes (addresses) of any one of 8 x 256 blocks from C0xx to C7xx. C7xx is the nominally selected block. The lowest 4 addresses (C700-C703) are not accessible as these are set aside to allow for de-selection of our peripheral devices.

Let's look at the 4 address lines. A0 and A1 are directed via buffers to the Bus. The changing of these lines controls the low order addresses. A4 is inverted, so while A4 is = 0 (inverted to a 1) and ANDed with the active C7xx output, A2 and A3 will be 0. By careful allocation of the 4 lines we can use 12 of the addresses out of the 16 available and this allows us to connect up to 3 PIA/6821s or 6 ACIA/6850s or even a 6840 timer or any combination of these devices. You can of course connect in just about any I/O device that requires 12 or less addresses. Fig 3 shows the connections of the address lines for the different devices.

This is how the general circuit works. A12-A15 are connected via inverters and a 4 input NAND gate to U4 which is a 74LS138. By arranging address lines onto the inputs of U4 (A8-A11) we can select any block of 256 addresses in the Cxxx range. Pin 6 of U2 will be low for any address in Cxxx thus enabling U4. C7xx is now ANDed with A2 and A3 and as previously explained provides the selects for our I/O Bus. C7xx is also ANDed with R/W and O2 clock. Pin 12 of U3 (a 7411) will go high which causes U6 and U7 (the Data Buffers) to allow the processor to READ from the bus under the control of the Signal DD which is generated by the inverter U1 and diode D1 (D1 gives an open collector condition to the bus from U1). Under conditions of non selection of C7xx or when R/W is 0 then the Data Buffers are writing to the I/O Bus, this has no effect unless an actual device is selected.

Fig 4 shows the logic requirements to implement a second 16 pin I/O output. And Fig 5a shows the requirements for a full 8 wide I/O connector on the 16 pin Bus, using NOR gates. Fig 5b shows a simplified circuit using a 3 to 8 decoder. The 8 ports are all within the same 256 Byte address block.



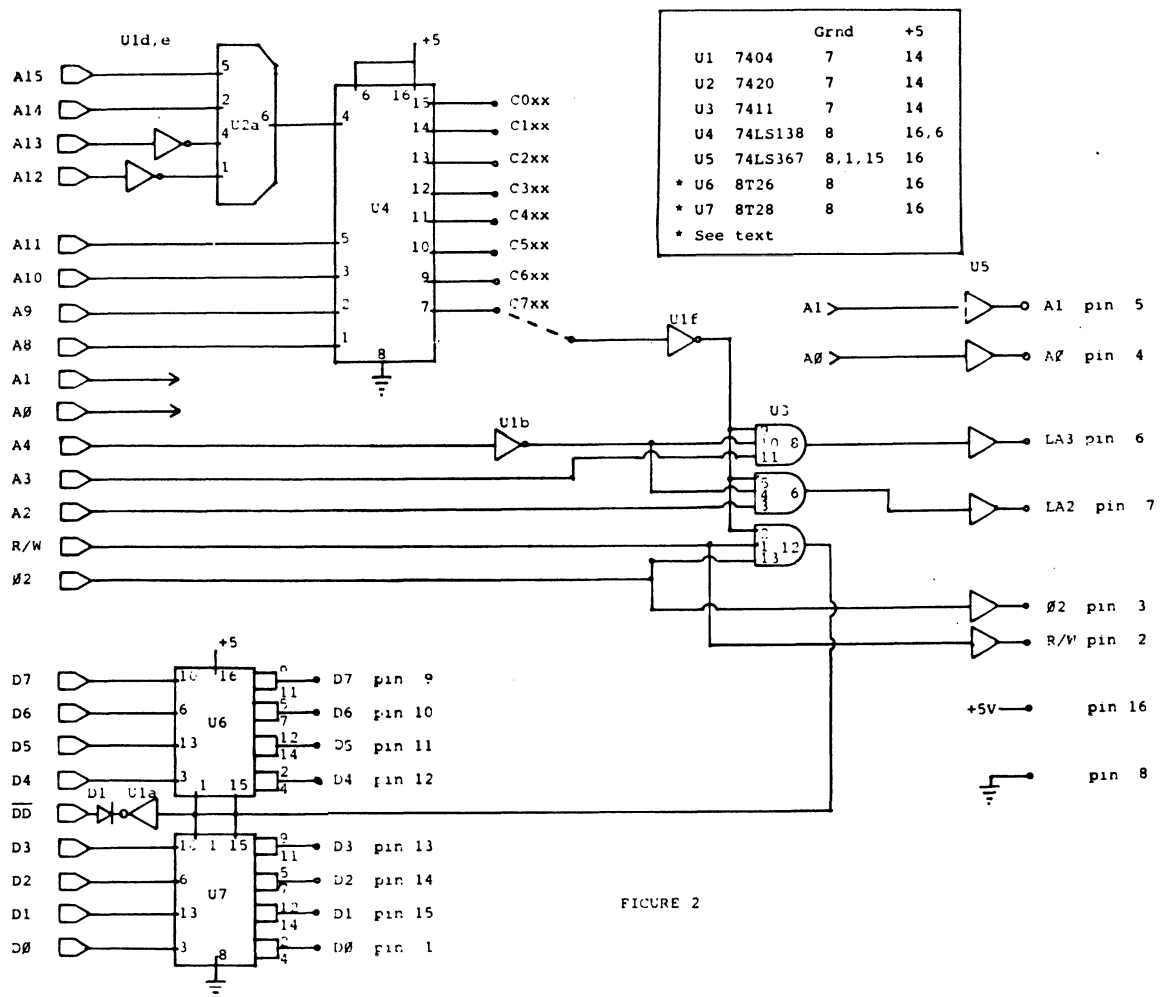


FIGURE 2

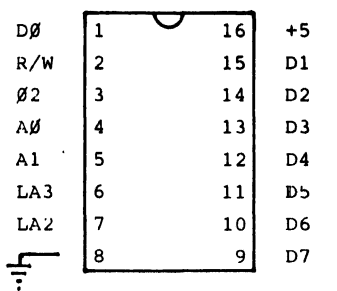



FIGURE 1

Connections to 16 pin  
DIP Header 16pin I/O Bus

Device	AØ	A1	LA2	LA3	Address	Misc. pins
6821-1 PIA	36	35	24	23	C7Ø4-C7Ø7	22 +5
6821-2 PIA	36	35	23	24	C7Ø8-C7ØB	22 +5
6821-3 PIA	36	35	22	24	C7ØC-C7ØF	23 Grnd
6850-1 ACIA	11	8	10	9	C7Ø4-C7Ø5	
6850-2 ACIA	11	8	10	9	C7Ø6-C7Ø7	
6850-3 ACIA	11	8	9	10	C7Ø8-C7Ø9	
6850-4 ACIA	11	8	9	10	C7ØA-C7ØB	
6850-5 ACIA	11	9	8	10	C7ØC-C7ØD	
6850-6 ACIA	11	8	8	10	C7ØE-C7ØF	
6840-1 Timer	10	11	12	16	C7Ø8-C7ØF	15 Grnd

FIGURE 3 - Address line connections

For a combination of devices just remove one type then add devices of another to fill up that address space.

A symbol  in an address box means invert that signal before connecting to pin number.

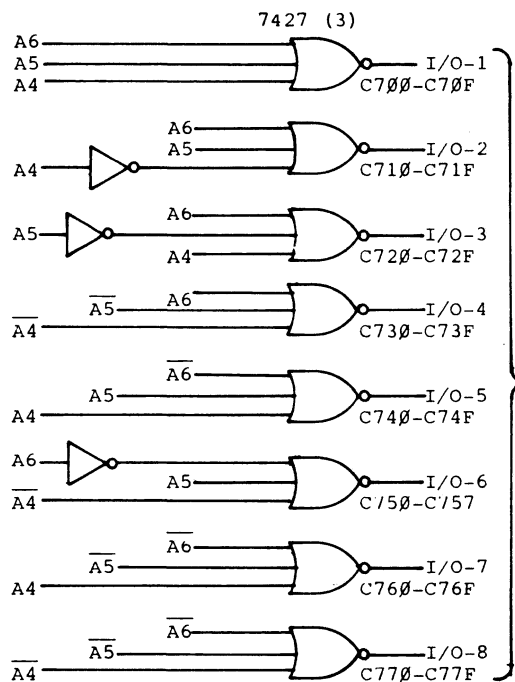


FIGURE 5a  
Using NOR gates as a 3/8 decoder

Connect to  
7411 where  
A4 used to  
be on each  
LA and gate

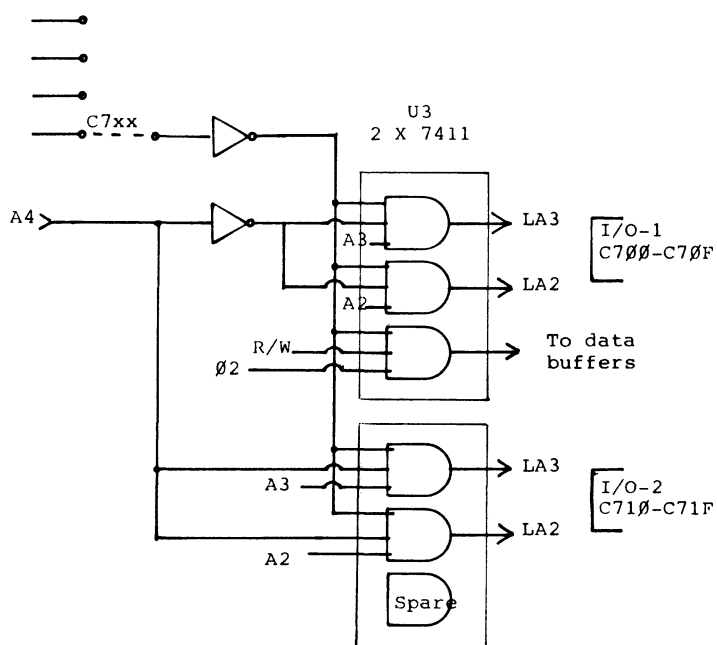


FIGURE 4 Simple extension to two 16 pin I/Os

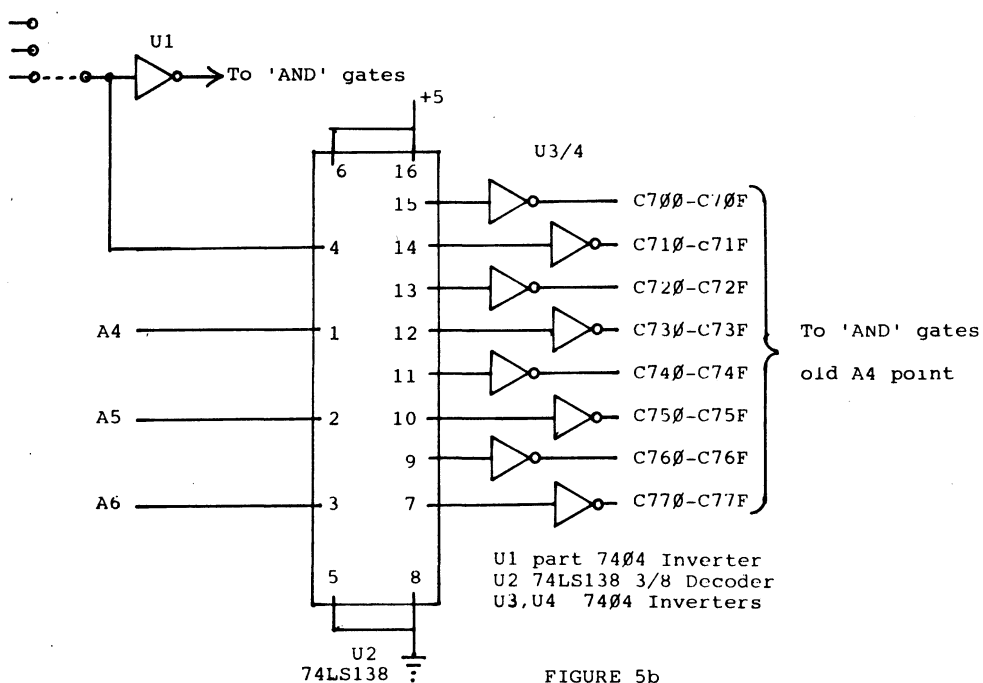


FIGURE 5b

## FLOATING POINT FOR FORTH

*by Adam Dickson*

One of the major aspects of FORTH which deters potential newcomers is the absence of floating point operations. Whilst a surprising amount of numerical work can be done without floating point (FP) i.e. using various fixed point systems, the fact remains that floating point is unsurpassed for such applications as graphics and mathematical simulations where very large numerical ranges are often involved.

Until now I have continued to use BASIC, not for any virtue inherent in the language as such, but solely because of its FP. Moreover, BASIC is extravagant. FP is used even if I want to manipulate individual bits, or % "integers". Slow!

My FORTH system is a standard fig model (FORTH TOOLS), running with a disk. To add FP to this language one could write the appropriate routines from scratch, either in assembler, CODE, or high level FORTH. This would be re-inventing the wheel.

I chose instead to use the 6.5 digit FP routines lying dormant in BASIC-in-ROM (\$A000-\$BFFF) and incorporate them into FORTH. These are fast and have a precision just right for graphics etc. Moreover I can still use integers where they are most suited. After much Q'ing of EXMON I found the basic two operand arithmetic functions.

If A,X,Y are the 6502 registers, P points to one of the FP operands and equals Y\*256+A, i.e. (P) is one of the operands (4 bytes long). The FP format is basically the usual "Normalised Binary Exponential", i.e.  $FP = X \cdot (2^Y)$  where X is a binary fraction ranging from .5 (.1000000000...) to .99999.. (.1111111111...) (the mantissa, the sign incorporated in the most sig. bit). Y is a signed integer (the exponent).

The format:

(P) = Exponent ( +\$80 as always).

(P+1) = Most sig. byte of normalised mantissa, with sign superimposed on bit 7.

(P+2) = Middle sig. byte

(P+3) = Least sig. byte

This is the standard variable format (in BASIC). Zero is unique in this system, and is marked by a zero exponent.

The other FP operand is stored in the primary floating point accumulator (FP1), in a different format, referred to as accumulator format. It is at \$AC-\$B0

\$AC - exponent +\$80

\$AD - Most significant byte

\$AE - Middle sig. byte

\$AF - Least sig. byte - OF MANTISSA (NORMALISED) i.e. bit 7 of MS byte is always a "1".

\$B0 - Sign in bit 7, 0 +ve 1 -ve

When these and other locations are set up, calling the following routines will perform the corresponding FP operation.

\$B46C  $FP = (P) + FP$

\$B455  $FP = (P) - FP$

\$B5FB  $FP = (P) * FP$

\$B6CA  $FP = (P) / FP$

Upon calling these routines the FP number at P is loaded to the secondary FP accumulator (FP2) \$B3-\$B7, the calculation is done and the result is left in the FP1 in the usual FP accumulator format. NOTE - There are other locations of importance (which must be set up)

\$B9 (bit 7) always holds a "25th BIT" (the least significant) of FP1 after calculations which is used for rounding. Such rounding is done only upon storing FP1 elsewhere (say, in BASIC as a variable, in variable format). Obviously it is set to zero, when loading FP1 (with a variable format FP number). There is no similar rounding bit for FP2, i.e. FP2 is always kept rounded (to a 24 bit mantissa).

\$B8 holds sign (FP1) ex-ored with sign (FP2), when FP2 is loaded at the beginning of a FP +-\* / operation (\$B64D).

\$71-\$75 are used for partial product/quotients in the \*/ operations.

There are further routines -

\$B74B moves a FP (variable format) starting at Y\*256+A into FP1 (translating into accumulator format) and sets up \$B9 to zero.

\$B778 moves (and translates into variable format) the primary accumulator, to memory starting at Y\*256+X, uses 25th bit to round the FP number being stored (\$B7BA).

There are also the complex FP I/O routines, which translate ASCII to normalised binary exponential and vice-versa.

\$B96E translates FP1 (destructively) into ASCII - including signs and exponential notation starting at \$0100, the end being marked with a null.

\$B887 using the CHRGET (\$00BC) routine and translates the ASCII FP number pointed to by (\$C4)\*\$100+(\$C3), (part of a LDA (ABS)) into the binary FP form, in FP1, in accumulator format. Upon calling, A must have first character, i.e. via CHRGET \$00C2.

This determined, it remained to hook this to FORTH. Due to the common memory usage, swapping of memory is done, not only to preserve the FORTH, but to make the FP routines think they are running BASIC. I am swapping over 80 bytes. Whilst this is more than needed, a reduction must wait for more detailed knowledge of the routine's memory usage. For the sake of speed I am converting to a hardware switch between two 0/1 page memory banks - this will speed up the DOS too.

What about error trapping? The only way to do this (without burning EPROMs) was to catch them via the print vector \$021A, \$021B - on the printing of the first character of an error message. Luckily X also gives an indication of the type of error at this point. This was used to print out OVERFLOW ERROR (X=\$A) and DIVIDE BY ZERO ERROR (X=\$14) as required.

There are two programs provided, a largely unoptimized assembly driver and 3 screens of FORTH, to hook the assembly code into FORTH and to introduce the statements. A fourth FORTH screen (48, imaging \$9D00 onwards) is used to hold the assembly object code.

The new FORTH words are:

```
F+    f1 f2 --- f1+f2
F-    f1 f2 --- f1-f2
F*    f1 f2 --- f1*f2
F/    f1 f2 --- f1/f2
F#    f --- address to print from    number of characters to type
F.    f --- prints out, where f, f1, f2 are variable format FP
      numbers on the stack.
FP    used to enter a FP number onto the stack, (see FORTH
      examples).
```

All FP numbers are entered and printed in standard BASIC format. The format of FP numbers on the FORTH parameter stack is:

```
BOT+3 : BOT+2 : BOT+1 : BOT
LS BYTE  MID BYTE  MS byte+sign  Exponent
              of mantissa
```

(BOT is \$00,X in 6502 FORTH)

The standard variable format of ROM BASIC is used. Since each FP number on the stack occupies 4 bytes (2 words), standard double word definitions, i.e. DVARIABLE, DDUP can be used. For consistency similar words with "F" instead of "D" could be defined.

What remains to be done? At present the FORTH routine QUITs in case of error. If one wants no breaks in execution one must put "infinity" (i.e. the largest possible number) of the appropriate sign instead of QUITing (for the basic arithmetic operations, at least).

Calling BASIC functions, such as SIN, INT, LOG, RND.  
FP to Integer, Integer to FP conversion. i.e. using \$AE88,\$AFC1.  
A hardware swap of page 0/1. Better still, to throw out all this and use an AMD9511,8087, or even a 16081 FPU.

These will be dealt with in relation to 3D graphics in coming issues.

An example of operation:

f(radius) --- area of circle printed out

```
: C 2DUP F* FP 3.1415926 F* F. ;  
FP 1.76 CR C  
9.73140 OK
```

NB. As with all literals, FP numbers in colon definitions will normally be compiled at once into binary (normalised exponential).

For more details on this, and on a complete floating point model written in FORTH (to emulate 9 digit BASIC), call me on

LATE NOTE: At the last minute I was able to include several BASIC functions (single operand, single result) in the listings (see screens 43,44). Their operation is self evident, and better error messages are provided. Note that these have not been checked for memory usage, hence are not fully debugged. Use with caution.

*Listings will be in next month's newsletter, or contact KAOS and they will post a copy to you.*

---

OSI HISTORY Part 2  
*by Eric Lindsay*

This technical inovativeness was not matched at the sales end, nor in support. OSI had large full color advertisements in the computer magazines, and for a long time were, on paper, arguably the best value for money, and had the fastest micro available. However relations with dealers never seemed to be all that good, and factory support seemed non-existent. There was a joke competition going the rounds for a word to describe OSI's "factory dis-support". The company rapidly gained a reputation for being impossible to talk to, and difficult to obtain information from.

Towards the end things did improve. OSI had always supplied a quantity of technical information with their equipment, and a limited, but useful, guide to the Basic. New workshop manuals were produced for them by Howard W. Sams, and these are among the best (and cheapest) such manuals available for any microcomputer. There were upgrades to the disc operating system, extensive new manuals, and a reasonable tutorial for new users. But it was too late to regain a reputation, too many dealers had turned to selling other, "easier"

products. OSI were taken over by M/A-COM. Silence prevailed, for many months, until a range of "new" models were announced. They looked good on paper, but I have no idea whether any were delivered, for OSI were sold again, to Kendata, a company only a fraction of the size of OSI. An even later rumour indicated that OSI were back in business, being run by their own employees.

It probably doesn't matter too much to the hobbyist user, although most hobbyists have a high regard for the quality of the equipment from OSI, for there are many substitutes. In the USA D&N Micro Products and other firms are making OSI compatible computers and boards. "Practical Electronics" magazine in England released the UK101, a slightly modified kit version of the OSI Superboard. "Elektor" designed a \$30 version of the OSI disc controller for use with their "Junior" computer, running the OSI disc operating system. In Australia several active user groups appeared, and designed many upgrades, and replacement bus systems. Recently complete expansion systems, and memory boards have become available. A totally redesigned OSI compatible computer became available, designed in Australia to use modern components. The Ozi Rabble comes as both an expansion board (32k RAM, 32k ROM, PIA, VIA, disc controller), and as a complete replacement for the OSI, able to run 5" or 8" disc drives. At the same time, the remains of OSI started selling their Superboard computers at prices as low as \$100, and expansion units for not much more.

OSI may now be dead for businesses, but for hobbyists, it remains one of the cheapest paths to a powerful computing system. It also offers far more opportunities to learn about how computers work - some people say it forces you to learn how they work. But while you can joke, you should also note that these 1977 vintage computers still benchmark faster than the latest IBM PC in Basic, that their interpreted BASIC can run as fast as the UCSD Pascal system on an Apple or TRS80, and that you can get one up and running for under \$200. You can expand that same system up through mini floppies, to floppies, multi tasking, and multiple processors. For an antique micro, from a "defunct" maker, and running a lowly 6502 games chip (albeit at clock speeds of up to 3.3 megahertz), they can still provide some impressive results.

*Continued next month.*

---

### RATBAS LISTING (continued)

```

5630 NEXT J
5640 PRINT "LINE"gt(i)": Undefined label."
5650 error+1
5660 GOTO5750:REM Do next goto
5690 *
5700 PRINT#f2,gt(i)"goto"gl(j)
5750 NEXT I
5760 REM Output label line code..
5770 IFcl=0THEN5800
5780 FORl=0TOcl-1
5790 PRINT#f2,gl(i)"rem":NEXT
5800 REM
6000 DISKclose,6
6008 REM The following line will return control to the
6009 REM Command file after loading..
6010 PRINT#f2,"disk!"CHR$(34)"io 10"CHR$(34)
6020 DISKclose,7
6100 PRINT#error"error":IFer<>1THENPRINT"s";
6110 PRINT" found."
6200 IFer=0THEN6400
6210 END
6390 *
6400 INPUT"Load program"ia$:a$=LEFT$(a$,1)
6410 IFa$<>"y"THEN6210
6500 DISK!"ME d000,d000":REM Yep, it's command file time
6510 cr$=CHR$(13):qu$=CHR$(34)
6520 PRINT#5,"neu"cr$;
6525 PRINT#5,"diskopen,6,"qu$"ratbas.in"qu$;cr$;
6527 PRINT#5,"disk!"qu$"io 20"qu$;cr$;:REM File ends with 'disk!'io 10"
6528 PRINT#5,"diskclose,6"cr$;
6530 PRINT#5,"diskopen,6,"qu$"ratbas.out"qu$;cr$;
6540 PRINT#5,"disk!"qu$"io 20"qu$;cr$;:REM File ends with 'disk!'io 10"
6542 PRINT#5,"diskclose,6"cr$;
6545 PRINT#5,"disk!"qu$"io 02"qu$;cr$;
6550 REM
6560 DISK!"io 10":END

```

FOR SALE ??  
*by John Whitehead*

A letter from a member saying that he did not know that Dabug was available for a 48x32 modified Superboard Series I, has prompted me to do the following:-

If you have for sale, Software or Hardware that you have designed, written or modified, that is not a one off item and will be available until at least the end of March 1984, then write to me at before the end of February 1984

Please supply details in the same format as in the example below.

ITEM	<i>Improved Extended Monitor in EPROM</i>
DESCRIPTION	<i>Contains 26 routines for machine code work including disassembling, block move, relocate and search.</i>
FULL DETAILS IN	<i>KAOS Sept 83</i>
WORKS ON	<i>SPI, SBII (SBII = Superboard Series II)</i>
MINIMUM RAM REQUIRED	<i>4K</i>
ANY EXTRAS NEEDED FOR ITEM TO WORK	<i>Any expansion board that will take a 2716 at \$E800 and a copy of the original Exmon manual.</i>
PRICE	<i>\$10 + \$1 P&amp;P</i>
AVAILABLE FROM	<i>Your name, address, phone number etc.</i>

Details of your produce will be printed in KAOS as soon as possible.

---

FOR SALE

C4P complete with 40K of CMOS memory on a Studio-Tech board,  
\$600.00 O.N.O.  
R. Kitto

32K Superboard (600 and 610 boards) 32 by 64 format (48 by 30 visible on screen)-C4 type keyboard, Professionally modified in Singapore. Cegmon style monitor ROM, plastic case including power supply, Modem and printer output plugs, fan. About 15 programs inc only 4 written for the correct display.  
\$400.00 O.N.O. Must be sold.  
Also KSR38 ASCII printer (15 inch tractor feed by Teletype Corporation). A very large machine. RS232 input. Works electronically but not mechanically. A bargain for any mechanical genius. Will sell to best offer.  
Paul Roehrs

SUPERBOARD Series 1, Dabug III, 8K RAM, power supply, manual, unpopulated 16K RAM board, populated 32X32 display board, Case with keyboard cutout, 30 programs on cassette.  
Price \$245.00 O.N.O.

COMP-SOFT is offering SKC 5.25" Floppy disks (100% certified) S/S D/D for \$36.00 a box. \$3.00 P&P

Model 15 Teletype complete with interface for connection to ACIA port, 120 page manual and driver software on cassette. \$50. Needs a 240V to 110V transformer. Software in EPROM can be arranged. John Whitehead.

Registered by Australia Post  
Publication No. VBG4212

If undeliverable return to  
KAOS, 10 Forbes St  
Essendon, Victoria 3040

K A O S K A O S  
K Postage K  
A Paid A  
O Essendon O  
S 3040 S  
K A O S K A O S